DS ASSIGNMENT NO: 05

Parenthesis Checker OR Syntax Parsing in Programming Languages

---

Name:- Aagam Gadiya PRN :- B24CE1118
Date:-

PROBLEM STATEMENT : **Parenthesis Checker:**

Write a program using a stack for push, pop, peek, and isEmpty operations. Write isBalanced() Function that Iterates through the input expression, Pushes opening brackets onto the stack. For closing brackets, it checks the top of the stack for a matching opening bracket. Ensures that all opening brackets are matched by the end of the traversal. Main Function: Accepts a string expression from the user. Uses isBalanced() to determine if the parentheses in the expression are balanced.

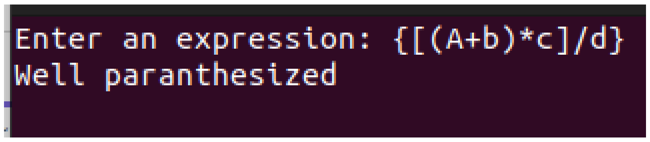
CODE

```
#include<iostream>
#include<string>
using namespace std;
class Stack {
   char arr[100];
   int top = -1;

public:


   void push(char c) {
      if (top == 99)
         cout << "Stack Overflow\n";
      else
         arr[++top] = c;
   }

   char pop() {
      if (isEmpty()) {
         cout << "Stack Underflow\n";
         return '\0';
```

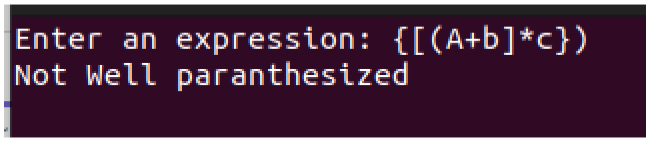
```
        } else {
            char pop_char = arr[top];
            top--;
            return pop_char;

        }
    }

    char peek() {
        if (isEmpty())
            return '\0';
        return arr[top];
    }

    bool isEmpty() {
        return top == -1;
    }
};


bool isBalanced(string expr) {
    Stack s;
    for (char ch : expr) {
        if (ch == '(' || ch == '{' || ch == '[') {
            s.push(ch);
        } else if (ch == ')' || ch == '}' || ch == ']') {
            if (s.isEmpty())
                return false;
            char top = s.pop();

            if ((ch == ')' && top != '(') ||
                (ch == '}' && top != '{') ||
                (ch == ']' && top != '[')) {
                return false;
            }
        }
    }
    return s.isEmpty();
}
```

```
int main() {
    string expression;
    cout << "Enter an expression: ";
    cin >> expression;

    if (isBalanced(expression))
        cout << "Well paranthesized" << endl;
    else
        cout << "Not Well paranthesized " << endl;

    return 0;
}
```

OUTPUT



```
Enter an expression: {[(A+b)*c]/d}
Well paranthesized
```



```
Enter an expression: {[(A+b]*c})
Not Well paranthesized
```